TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanksystemen* im SoSe18

Alexander van Renen, Maximilian E. Schüle (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss18/impldb/

Blatt Nr. 06

Hausaufgabe 1 Fensterln mit SQL

Analysieren wir die Gehälter von Professoren mittels Windowfunctions und führen Sie die Abfragen unter `hyper-db.de` aus. Dazu orientieren wir uns an der Relation *Professoren* des erweiterten Universitätsschemas:

```
with Professoren (persnr, name, rang, raum, gehalt, steuerklasse) as (
  values (2125,'Sokrates','C4',226,85000,1),
  (2126,'Russel','C4',232,80000,3),
  (2127,'Kopernikus','C3',310,65000,5),
  (2128,'Aristoteles','C4',250,85000,1),
  (2133,'Popper','C3',52,68000,1),
  (2134,'Augustinus','C3',309,55000,5),
  (2136,'Curie','C4',36,95000,3),
  (2137,'Kant','C4',7,98000,1)
)
```

1. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren.

   ```
   SELECT *, avg(gehalt) OVER () FROM Professoren;
   ```

2. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren partitioniert nach Rang.

   ```
   SELECT *, avg(gehalt) OVER (PARTITION BY rang) FROM Professoren;
   ```

3. Ermitteln Sie nun die wachsende Summe (das Quantil) des Gehaltes aller Professoren partitioniert nach Rang und absteigend sortiert nach ihrem Gehalt. Gleich verdienende Professoren sind im selben Quartil.

   ```
   SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC)
   FROM Professoren;
   -- aequivalent zu
   SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
   RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM Professoren;
   ```

4. Ermitteln Sie nun die wachsende Summe des Gehaltes aller Professoren partitioniert nach Rang und absteigend (total) sortiert nach ihrem Gehalt (reihenweise, nicht als Range-Query).

   ```
   SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
   ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM Professoren;
   ```

5. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus genau zwei mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang.

   ```
   SELECT *, avg(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
   ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING) FROM Professoren;
   ```

6. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus den 500 Einheiten mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang.
```
SELECT *, avg(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
RANGE BETWEEN 500 PRECEDING AND 500 FOLLOWING) FROM Professoren;
```
7. Geben sie zu jedem Professor das Gehalt des eins besser wie eins schlechter verdienenden.
```
SELECT *, lag(gehalt) OVER (ORDER BY gehalt DESC),
  lead(gehalt) OVER (ORDER BY gehalt DESC) FROM Professoren;
```
8. Ermitteln Sie die drei bestverdienendsten Professoren einmal mit und einmal ohne Windowfunctions.
```
SELECT * FROM (
   SELECT *, rank() OVER (ORDER BY gehalt desc) FROM Professoren
) WHERE rank < 4
```
```
SELECT * FROM professoren p_selbst WHERE 3>(SELECT count(*)
FROM professoren p_reich WHERE p_selbst.gehalt < p_reich.gehalt)
```

Hausaufgabe 2 Lösen Sie folgende Anfrage mit SQL basierend auf dem bekannten Universitätschema.

1. Bestimmen Sie die Durchschnittsnote für jeden Studenten.
2. Basierend auf dieser Durchschnittsnote, bestimmen Sie für alle Studenten ihren Rangplatz innerhalb ihrer Kohorte (Studenten desselben Semesters).
3. Berechnen Sie zusätzlich für jeden Studenten auch noch die **Abweichung** seiner Durchschnittsnote von der Durchschnittsnote der Kohorte (also vom Durchschnitt der Durchschnittsnote der Studenten der Kohorte) ausgegeben werden.

Lösen Sie Teilaufgaben 2 und 3 jeweils einmal mit und einmal ohne Nutzung von Windowfunktionen. Ihre Anfragen können Sie auf `hyper-db.de` testen. Nutzen Sie folgende erweiterte *pruefen* Relation:

```
with mehr_pruefen(MatrNr,VorlNr,PersNr,Note) as (
    select * from pruefen
    union
    values (29120,0,0,3.0), (29555,0,0,2.0), (29555,0,0,1.3), (29555,0,0,1.0)
    )
```

Ohne Windowfunktionen:

```
with mehr_pruefen(MatrNr,VorlNr,PersNr,Note) as (
select * from pruefen
union
values (29120,0,0,3.0), (29555,0,0,2.0), (29555,0,0,1.3), (29555,0,0,1.0)
),
noten(MatrNr,Semester,Note) as (
    select s.matrnr, semester, avg(Note)
    from studenten s, mehr_pruefen p
    where s.matrnr=p.matrnr
    group by s.matrnr,semester
)
select *,
(select count(*)+1 from noten x
    where x.Semester=n.Semester and x.Note<n.Note) as Rang,
(select avg(x.Note) from noten x
    where x.Semester=n.Semester) as GPA,
(select avg(x.Note) from noten x
    where x.Semester=n.Semester) - note as Abweichung
from noten n order by semester, rang
```

Mit Windowfunktionen:

```
with mehr_pruefen(MatrNr,VorlNr,PersNr,Note) as (
select * from pruefen
union
values (29120,0,0,3.0), (29555,0,0,2.0), (29555,0,0,1.3), (29555,0,0,1.0)
),
noten(MatrNr,Semester,Note) as (
    select s.matrnr, semester, avg(Note)
    from studenten s, mehr_pruefen p
    where s.matrnr=p.matrnr
    group by s.matrnr,semester
)
select *,
rank() over (partition by Semester order by Note asc) as Rang,
avg(Note) over (partition by Semester) as GPA,
avg(Note) over (partition by Semester) - note as Abweichung
from noten order by semester, rang
```

Hausaufgabe 3 Größer Fensterln mit SQL in TPC-H

Wenden wir nun Fensterfunktionen an dem fiktiven TPC-H Schema an.

1. Erstellen Sie in SQL eine Abfrage nach dem jährlichen Exportvolumen pro Jahr und Land. Verwenden Sie diese Abfrage als Hilfstabelle (`with`-Statement) in den folgenden Abfragen, orienteren Sie sich an TPC-H Anfrage 7.

```
with exportvolume as (
  select n_name, l_year, sum(volume) as revenue
  from (
     select n_name, extract(year from l_shipdate) as l_year,
        l_extendedprice * (1 - l_discount) as volume
     from supplier,lineitem,orders,nation
     where s_suppkey = l_suppkey
        and o_orderkey = l_orderkey and s_nationkey = n_nationkey
      ) as shipping
  group by n_name, l_year
)
```

2. Ranken Sie Länder anhand ihres jährlichen Exportvolumens, auf Platz eins ist das Land mit dem höchsten Volumen, das je in einem Jahr getätigt worden ist.

   ```
   select *, rank() over (order by revenue) from exportvolume;
   ```

3. Küren Sie nun die Jahressieger. Ranken Sie dazu die Länder partitioniert nach Jahr.

   ```
   select * from (
     select *, rank() over (partition by l_year order by revenue) from exportvolume
   where rank=1;
   ```

4. Ermitteln Sie nun das laufende Exportmittel der Länge drei (Vorjahr, Nachjahr, falls erfasst).

   ```
   select *, avg(revenue) over (
     partition by n_name order by l_year
     range between 1 preceding and 1 following)
   from exportvolume
   ```

Hausaufgabe 4

Die in Abbildung 1 dargestellten Relationen Mietspiegel und Kindergarten dienen der Bewertung von Wohngegenden im Großraum München. Für eine junge Familie ist ausschlaggebend, wie hoch die Lebenshaltungskosten gemessen an zu zahlender Miete und zu entrichtender Gebühr für den Kindergarten im jeweiligen Wohnort ausfallen. Illustrieren Sie die Ausführung einer Top-1-Berechnung (zur Bestimmung des günstigsten Wohnorts) für eine junge Familie mit zwei Kindern. Zeigen Sie die phasenweise Berechnung des Ergebnisses jeweils mit dem Threshold- und dem NRA-Algorithmus.

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

WohnLage	
Ort	Lage
Grünwald	München-Süd
Unterföhring	München-Nord
Ismaning	München-Nord
Garching	München-Nord
Bogenhausen	München-City
Nymphenburg	München-City

Abbildung 1: Münchner Wohnlagen zur Berechnung der monatlichen Kosten für eine Familie.

Siehe Lösungsbuch

Hausaufgabe 5

Geben die Relation Klausur:

MatrNr	Vorbereitungszeit	Note
1	150	1.7
2	70	2.7
3	450	2.0
4	180	1.7
5	2500	1.3

a) Formulieren Sie die Anfrage, die die MatrNr in der Skyline für die Attribute Vorbereitungszeit und Note erzeugt (kleiner ist jeweils besser) in SQL mit Hilfe des Skyline Operators.

b) Formulieren Sie die Anfrage in SQL ohne Skyline Operator.

c) Bestimmen Sie das Ergebnis der Anfrage.

```
with Klausur (MatrNr, Vorbereitungszeit, Note) as(
  values (1,150,1.7),(2,70,2.7),(3,450,2.0),(4,180,1.7),(5,2500,1.3)
)
```

SQL mit Skyline:

```
select MatrNr from Klausur k skyline of k.Vorbereitungszeit min, k.Note min
```

SQL ohne Skyline:

```
select MatrNr from Klausur k
where not exists (
   select * from klausur dom
   where
      dom.Vorbereitungszeit <= k.Vorbereitungszeit and
      dom.Note <= k.Note and (
         dom.Vorbereitungszeit < k.Vorbereitungszeit or
         dom.Note < k.Note)
)
```

Ergebnis:

1) Ist in Skyline (Kann in Vorbereitungszeit nur von MatrNr 2 dominiert werden, dort ist aber Note schlechter)

2) Ist in Skyline (Minimum für Vorbereitungszeit)

3) Ist nicht in Skyline, dominiert von MatrNr 1

4) Ist nicht in Skyline, dominiert von MatrNr 1

5) Ist in Skyline (Minimum für Note)

Hausaufgabe (wird nicht in der Übung besprochen) Cubes mit Studenten

Erstellen Sie anhand des Universitätsschemas einen Cube mit der Faktentabelle *mehr_pruefen* und den Dimensionstabellen *Vorlesungen, Professoren, Studenten*. Zählen Sie das Auftreten.

```
with mehr_pruefen(MatrNr,VorlNr,PersNr,Note) as (
   select * from pruefen
   union
   values (28106,5041,2137,3.0)
)

create table hoerentotal (vtitel varchar(30), pname varchar(30),
   sname varchar(30), all integer);

insert into hoerentotal(
   select v.titel as VTitel, p.Name as PName, s.Name as SName,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by cube(v.titel, p.Name, s.Name)
);
```

Ohne Cube-Operator entspricht die Abfrage bei d Dimensionen einer Vereinigung aus 2^d Abfragen (und somit $2^d - 1$ mal „UNION“), in diesem Fall bei drei Dimensionen acht Teilabfragen. Dabei geht man am besten strukturiert vor:

i	$[i]_2$	VTitel	PName	SName
0	000	0	0	0
1	001	0	0	1
2	010	0	1	0
3	011	0	1	1
4	100	1	0	0
5	101	1	0	1
6	110	1	1	0
7	111	1	1	1

Jedes Bit sagt aus, ob nach dem entsprechenden Attribut gruppiert wird oder nicht. Wenn danach nicht gruppiert wird, so ist der entsprechende Wert auf `null` zu setzen und bei Zahlen zu casten (`to_number(null)`).

```
create table hoerentotal (vtitel varchar(30), pname varchar(30),
   sname varchar(30), all integer);

insert into hoerentotal(
   select null as VTitel, null as PName, null as SName,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   -- keine Gruppierung
 UNION
   select null, null, s.Name as SName,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by s.Name
 UNION
   select null, p.Name as PName, null,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by p.Name
 UNION
   select null, p.Name as PName, s.Name as SName,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by p.Name, s.Name
 UNION
   select v.titel as VTitel, null, null
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by v.titel
 UNION
   select v.titel as VTitel, null, s.Name as SName,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by v.titel, s.Name
 UNION
   select v.titel as VTitel, p.Name as PName, null,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by v.titel, p.Name
 UNION
   select v.titel as VTitel, p.Name as PName, s.Name as SName,
      count(*) as all
   from mehr_pruefen pr, Vorlesungen v, Professoren p, Studenten s
   where p.PersNr = pr.PersNr AND s.MatrNr=pr.MatrNr AND v.VorlNr=pr.VorlNr
   group by v.titel, p.Name, s.Name
);
```